

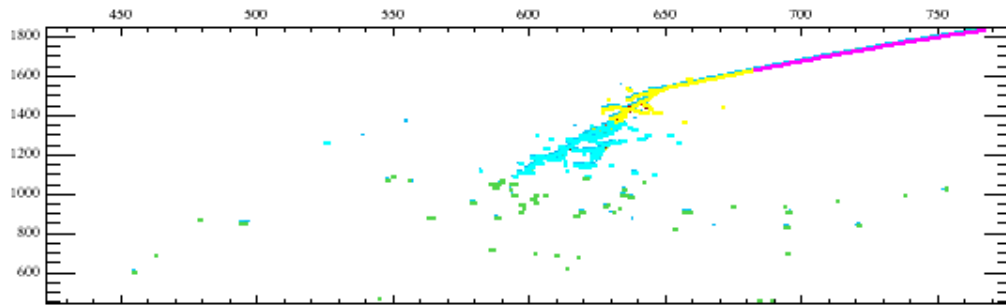
Update on fuzzy clustering

Ben Carls

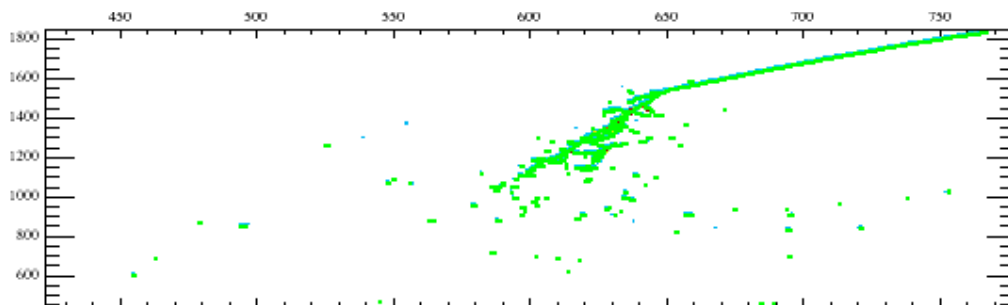
Status as of the last update

- Implemented a faster version of the Hough transform (PPHT)
- Still needed to acquaint the PPHT to the algorithm
- Need to deal with fake Hough lines found
- Had performed some code cleanup, created a base class for the normal Hough transform and the PPHT

Start with fuzzy clustering



Merge the clusters



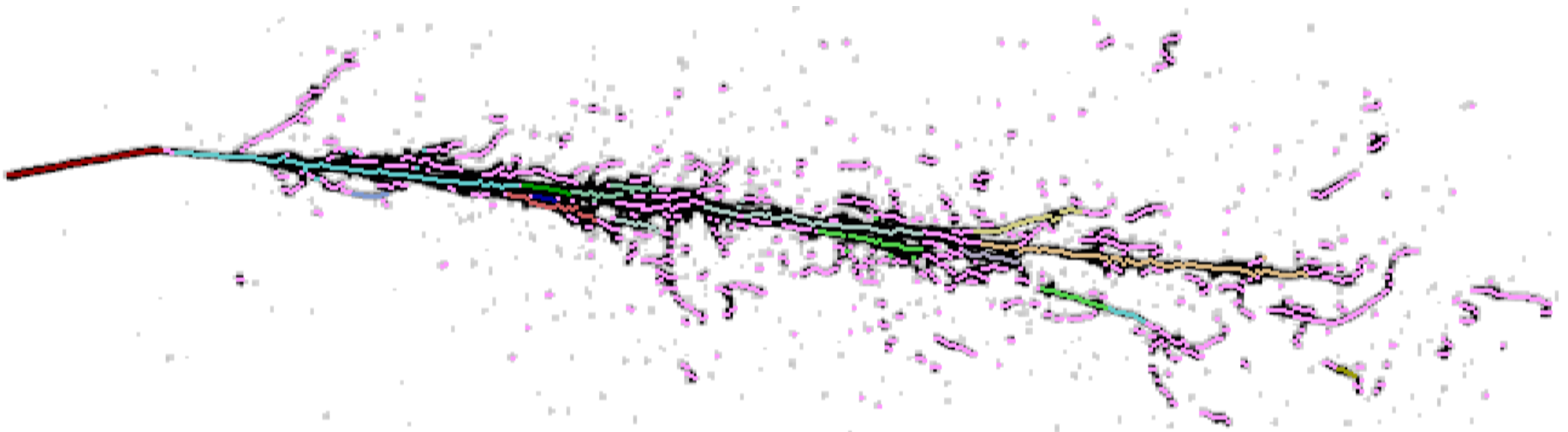
Fuzzy clustering assigns degrees of belonging, instead of definite belonging

Number of clusters determined using an optimization criteria (Xie-Beni index)

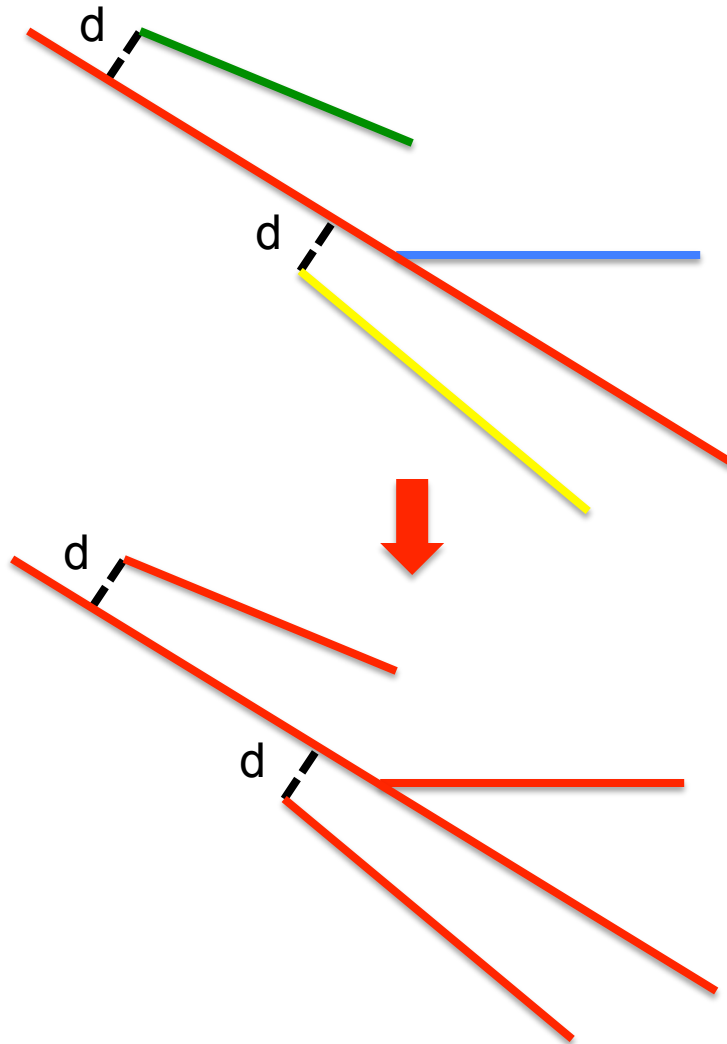
After running, the clusters are merged together

Hough line finder

- Run the Hough line finder to identify tracks and showers



Merging Hough lines

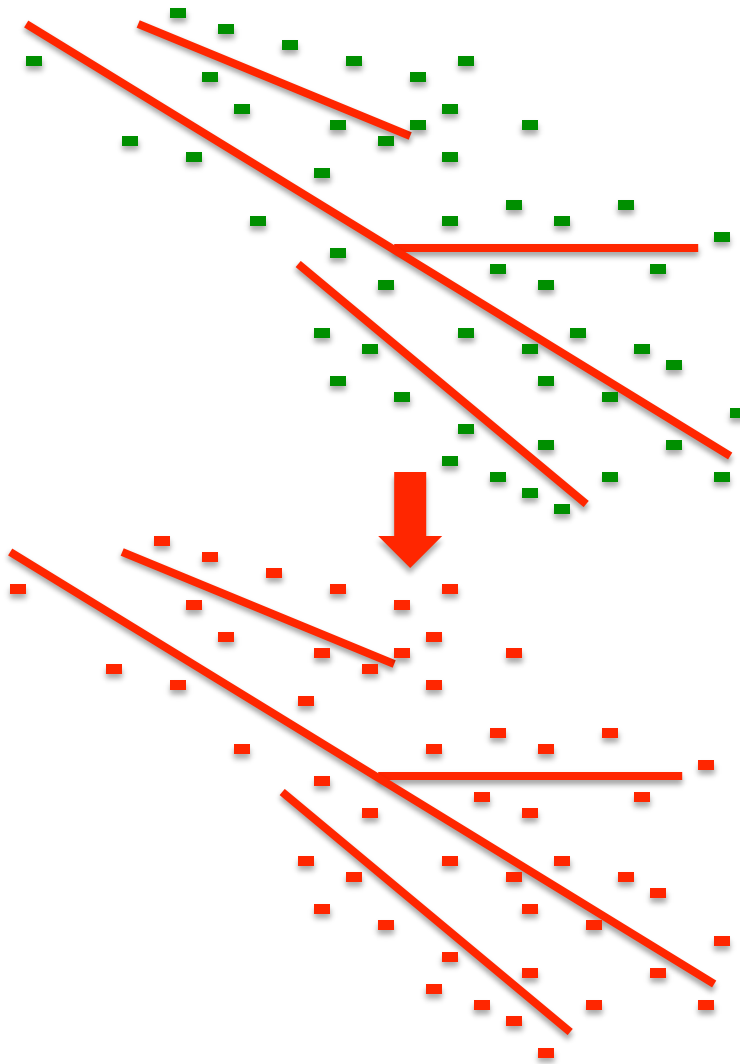


Merge Hough lines together to construct showers and tracks

Hough line finder very effective at finding lines in showers

Merge Hough lines using the distance between the line segments and angle between slopes $< 30^\circ$

The fuzzy cluster remnants



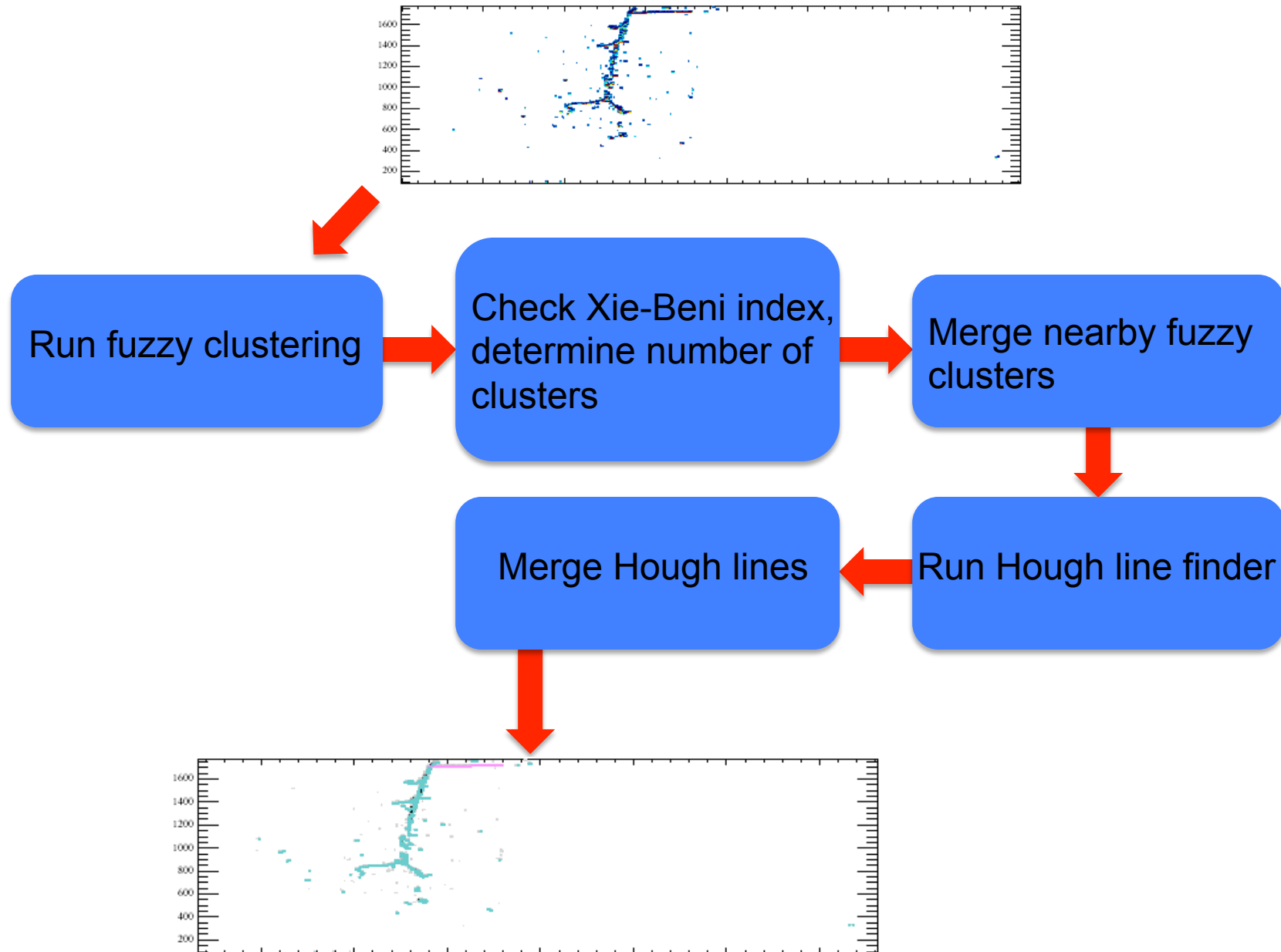
The issue remains with what to do with the left over fuzzy cluster points

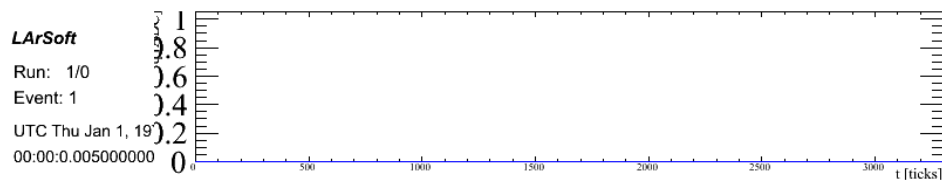
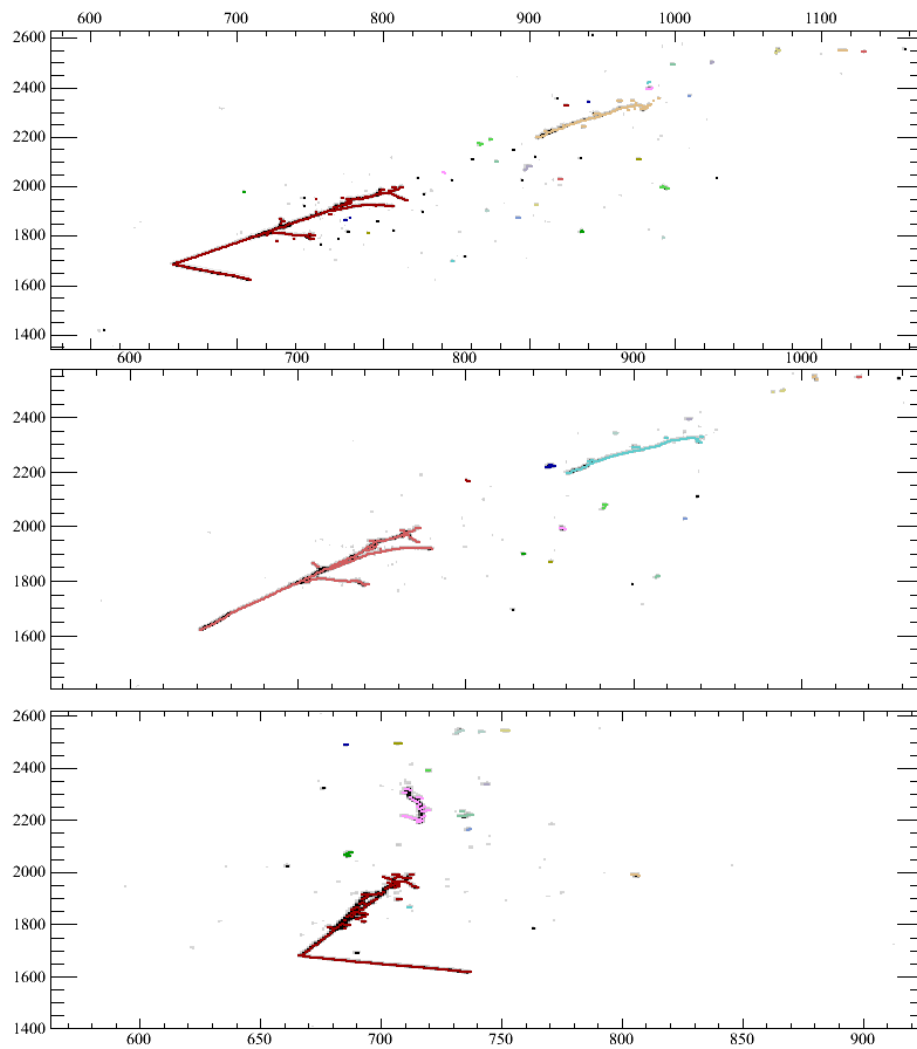
Currently merging these points into the nearest Hough line segment using:

$$d_{\text{weighted}} = \frac{\text{distance point to line segment}}{\text{length of line segment}}$$

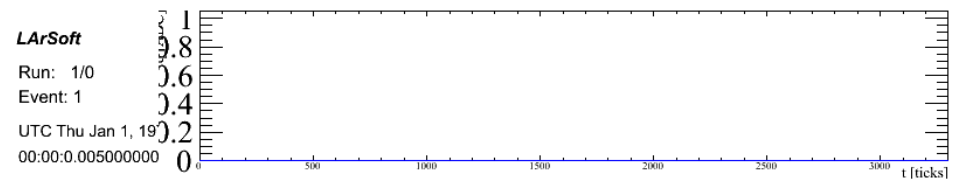
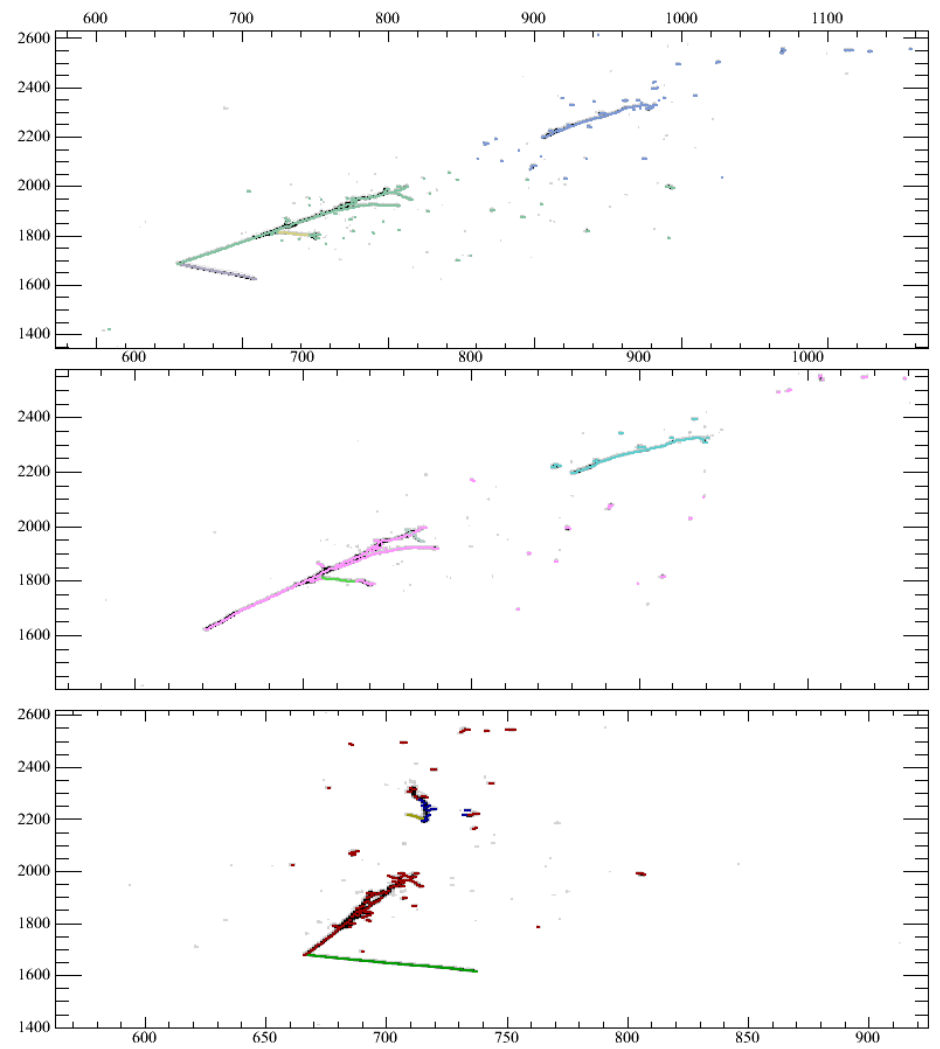
Goal is to give longer line segments more weight

Overview of the algorithm

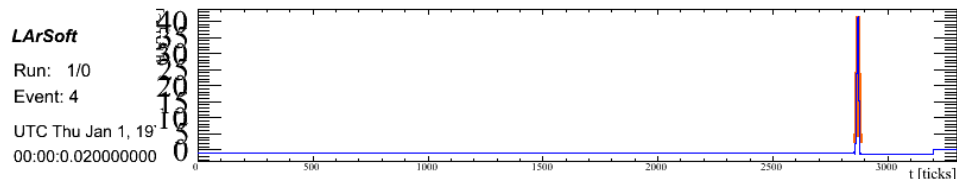
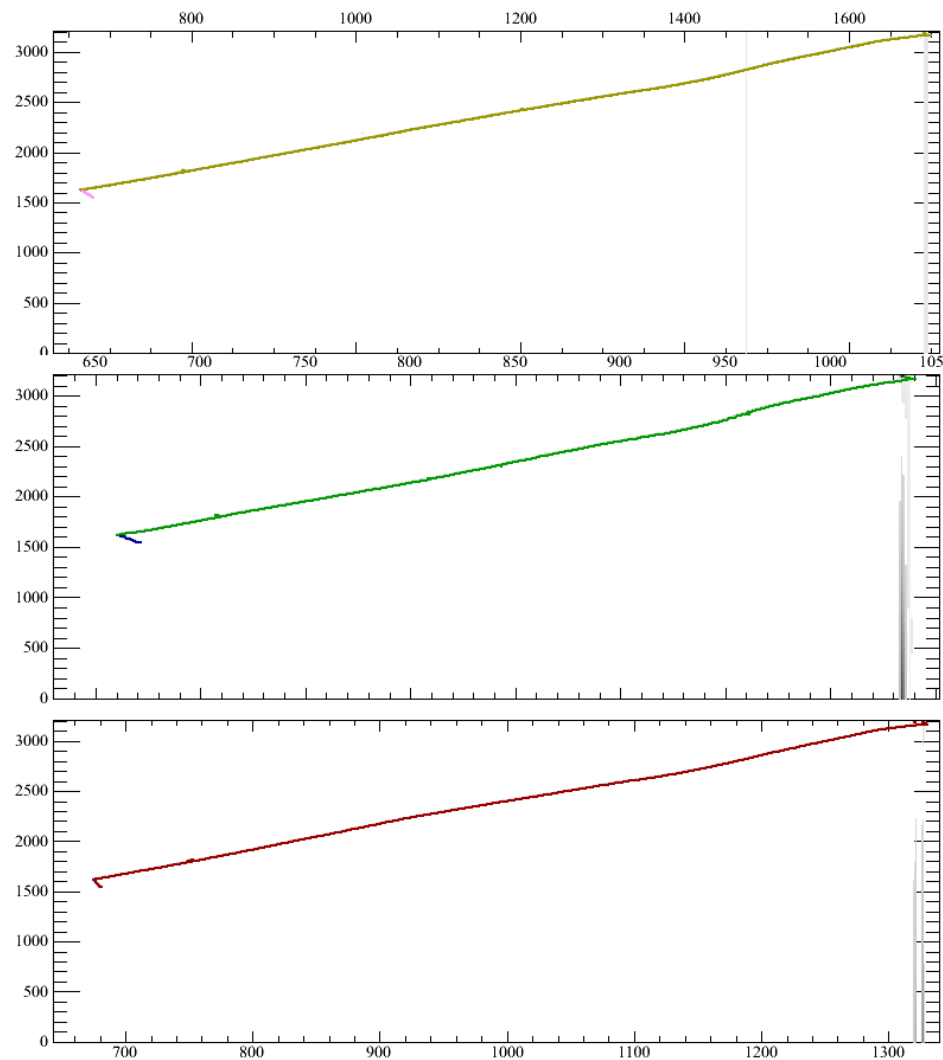
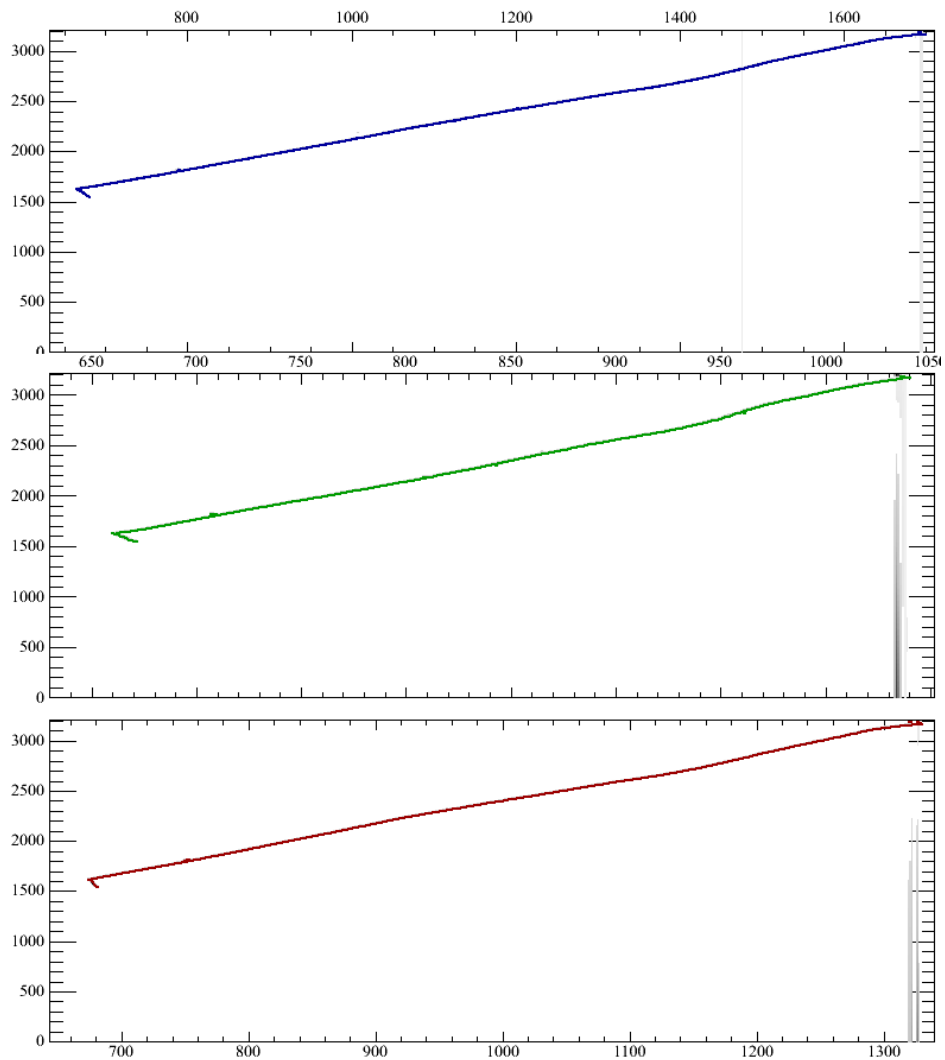




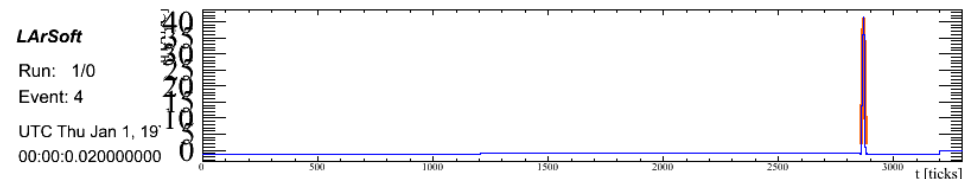
DBSCAN clustering



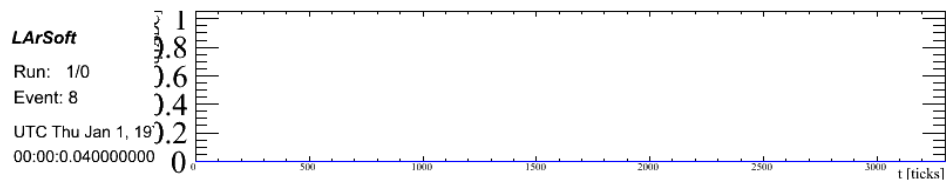
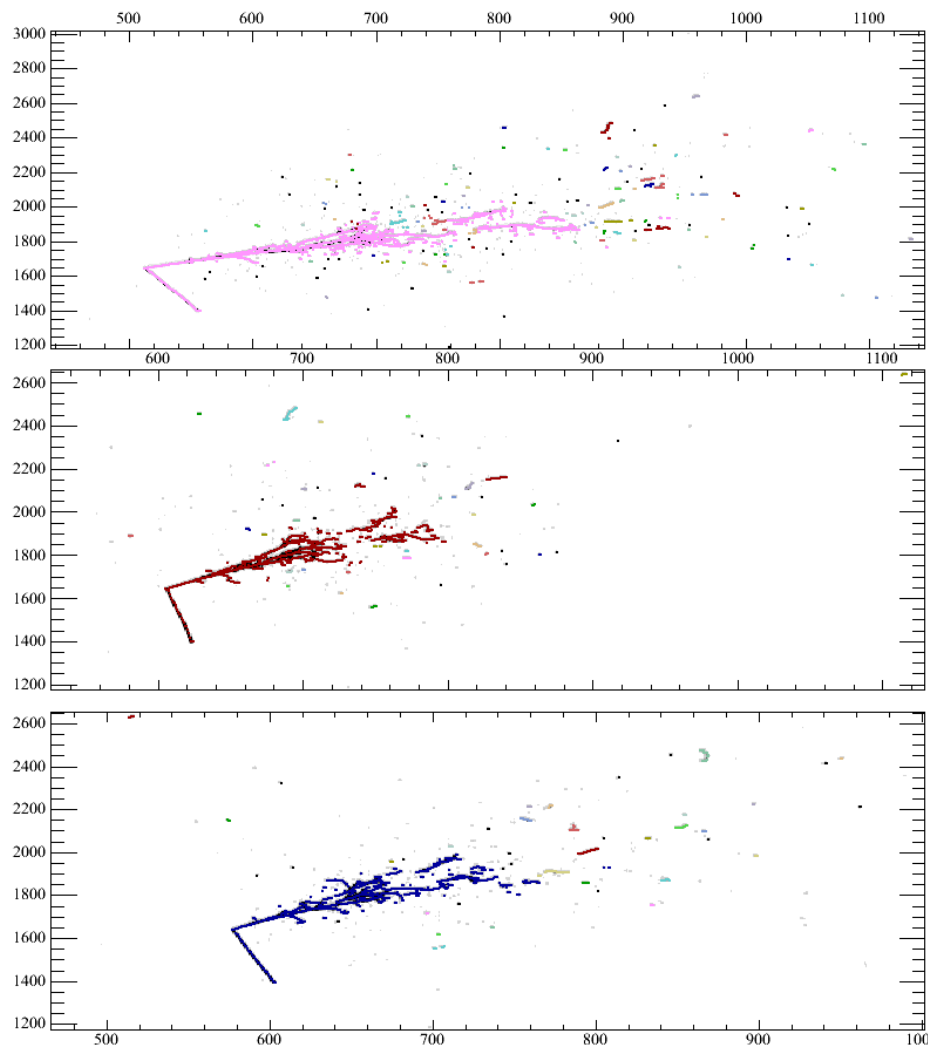
Fuzzy clustering



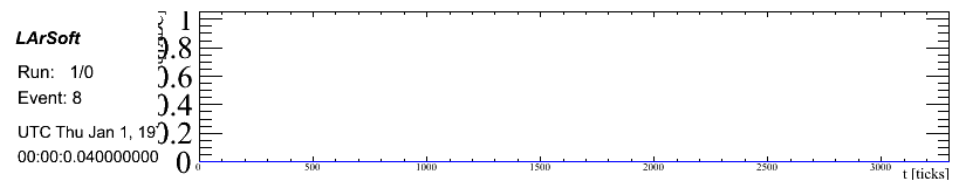
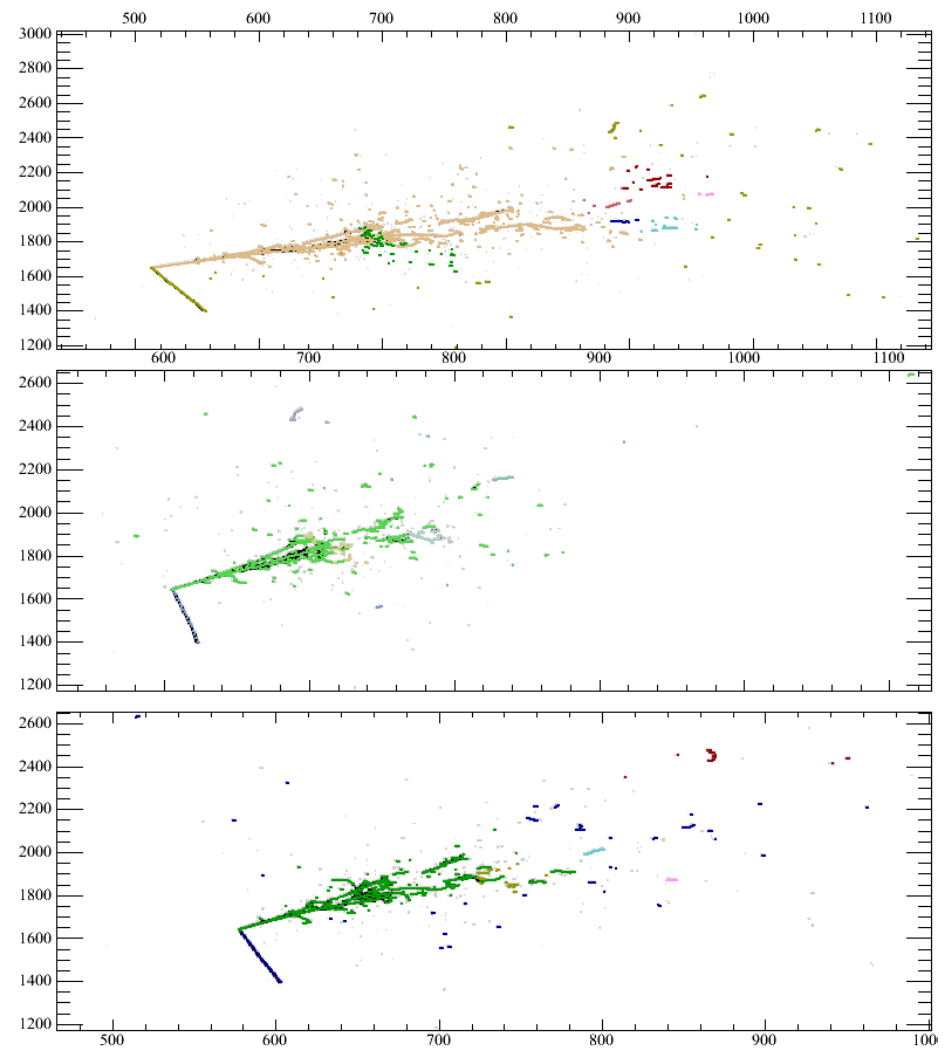
DBSCAN clustering



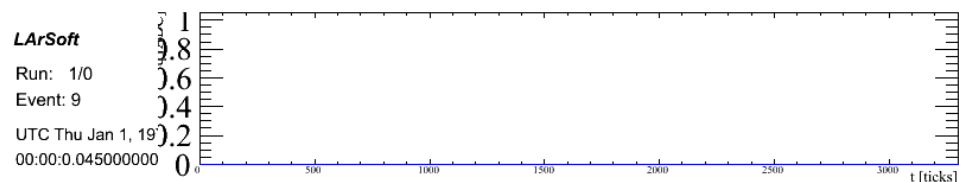
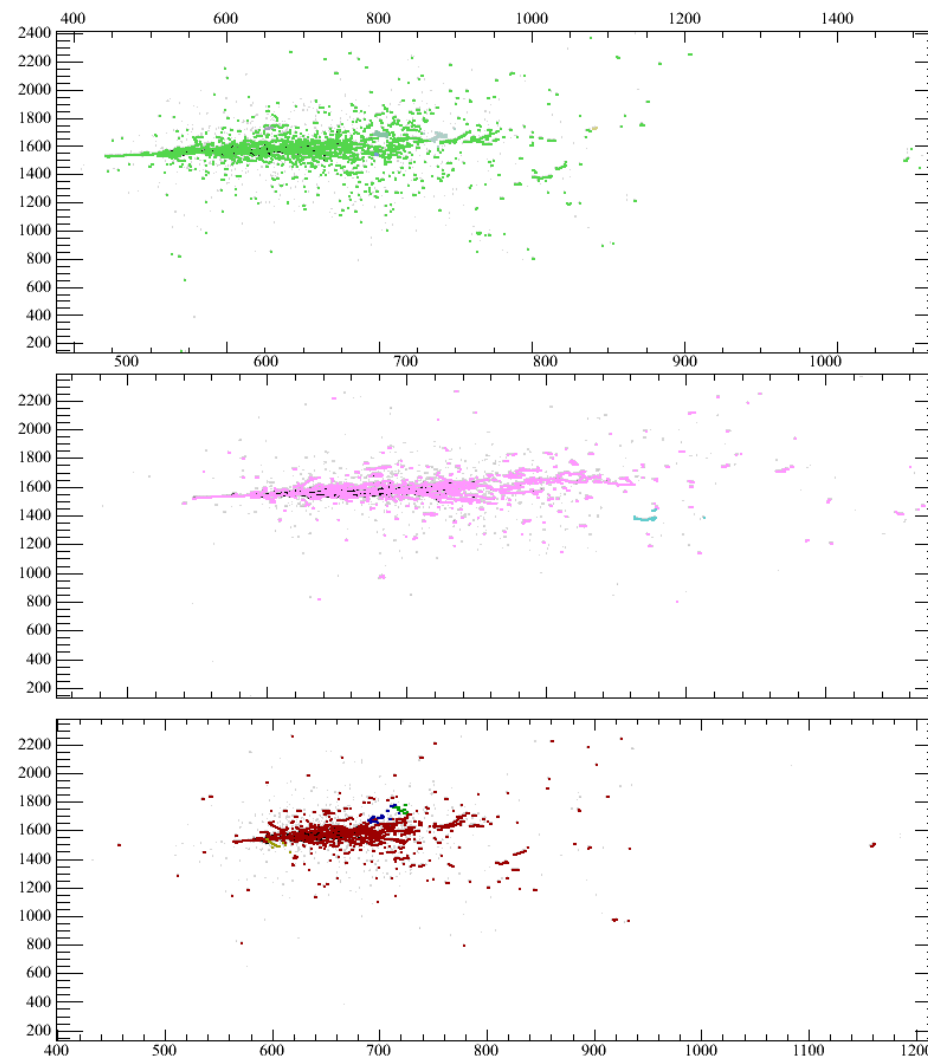
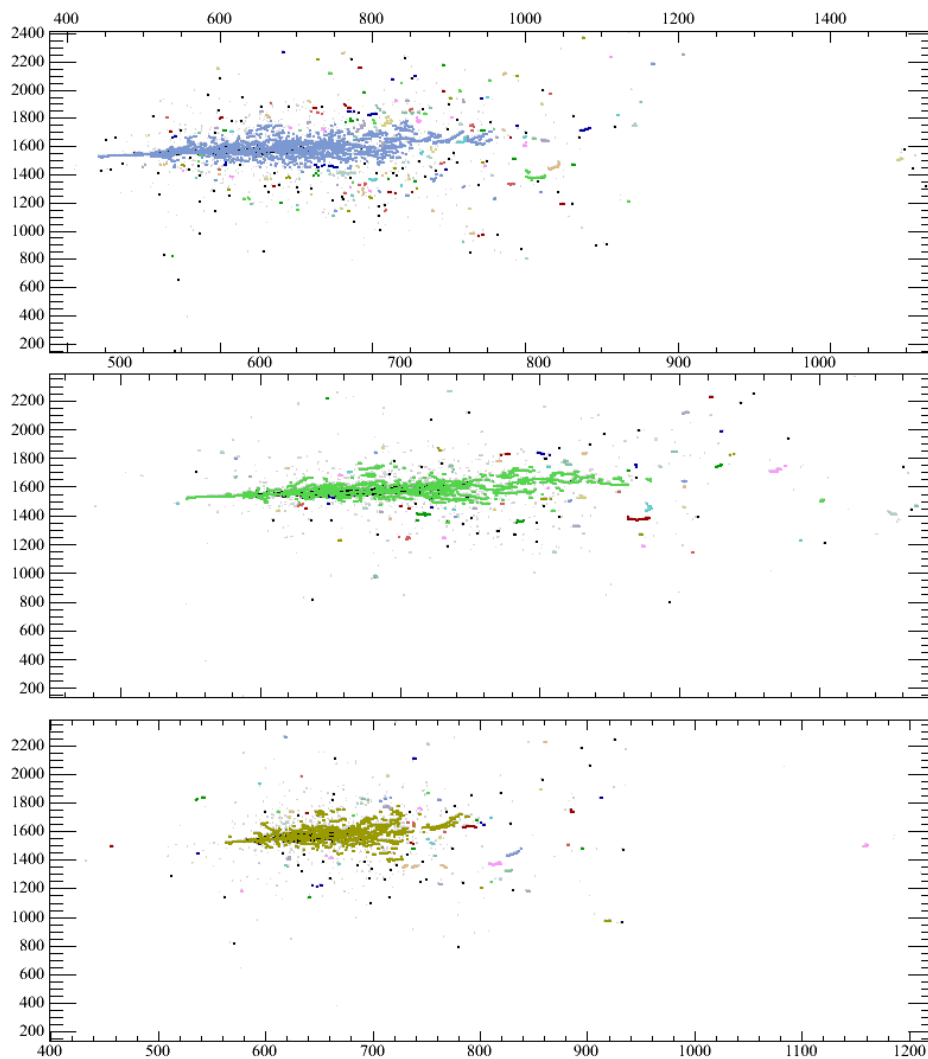
Fuzzy clustering



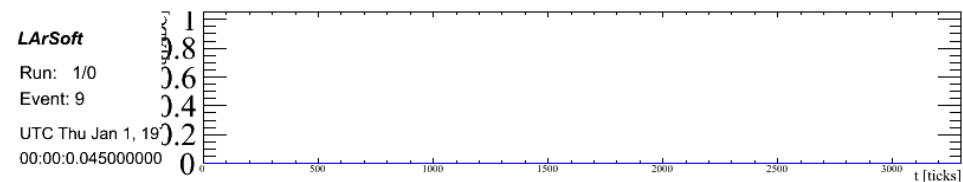
DBSCAN clustering



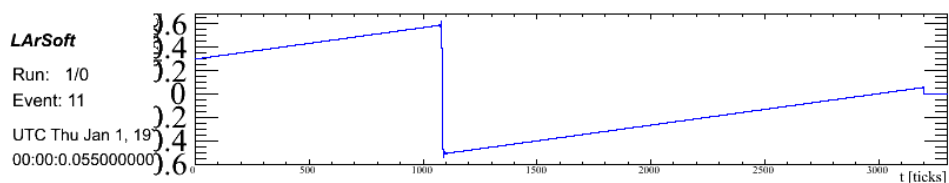
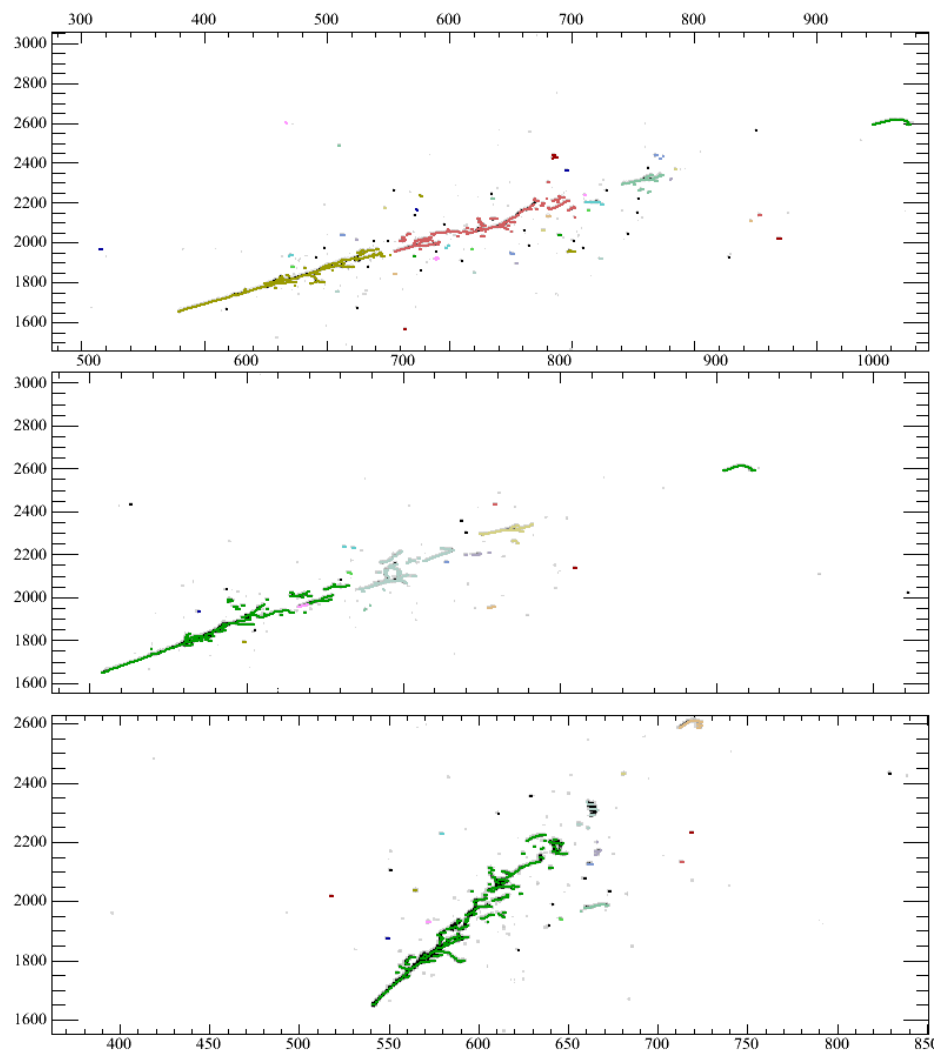
Fuzzy clustering



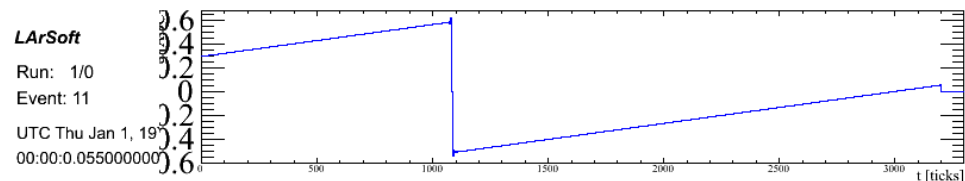
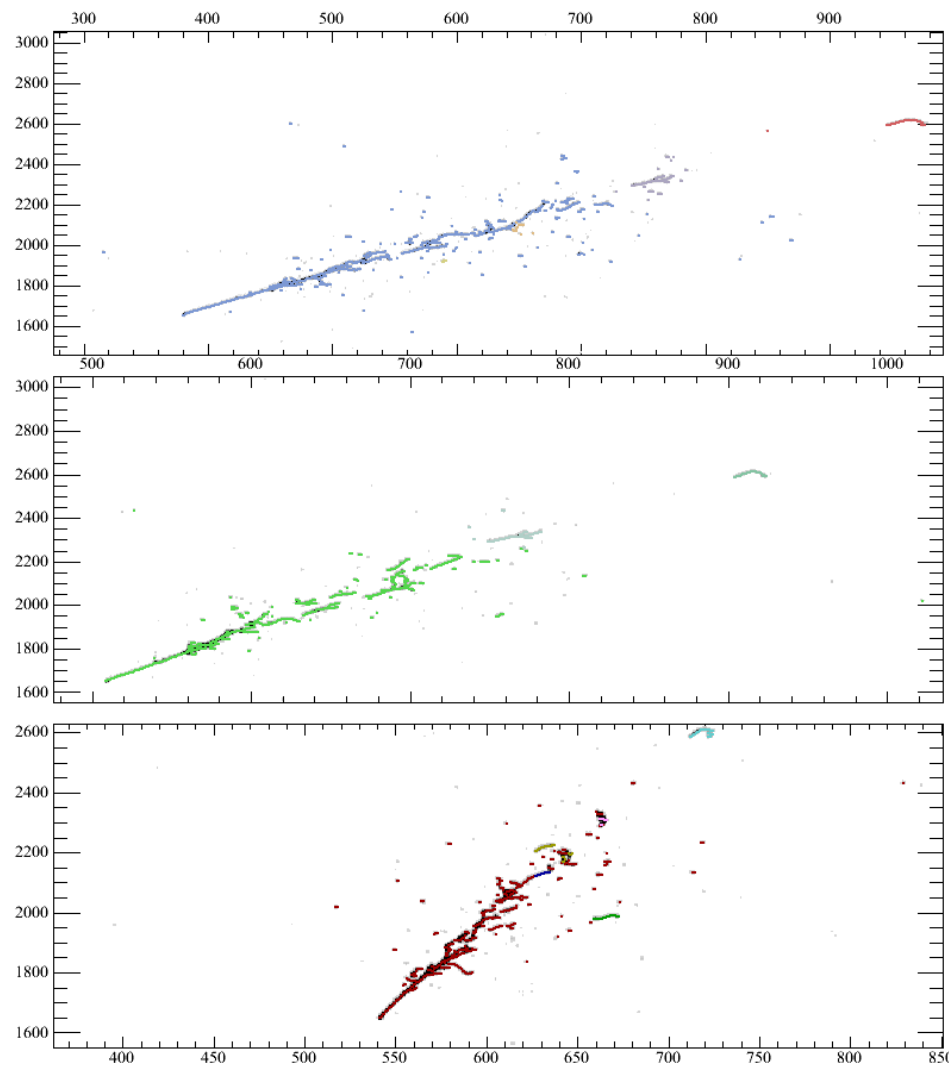
DBSCAN clustering



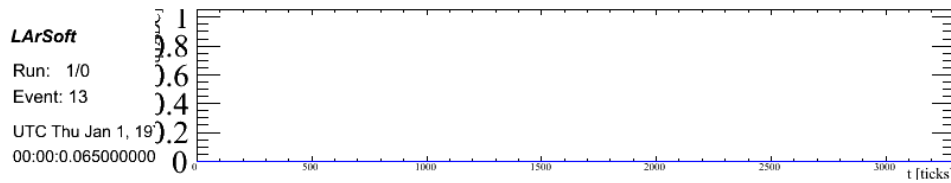
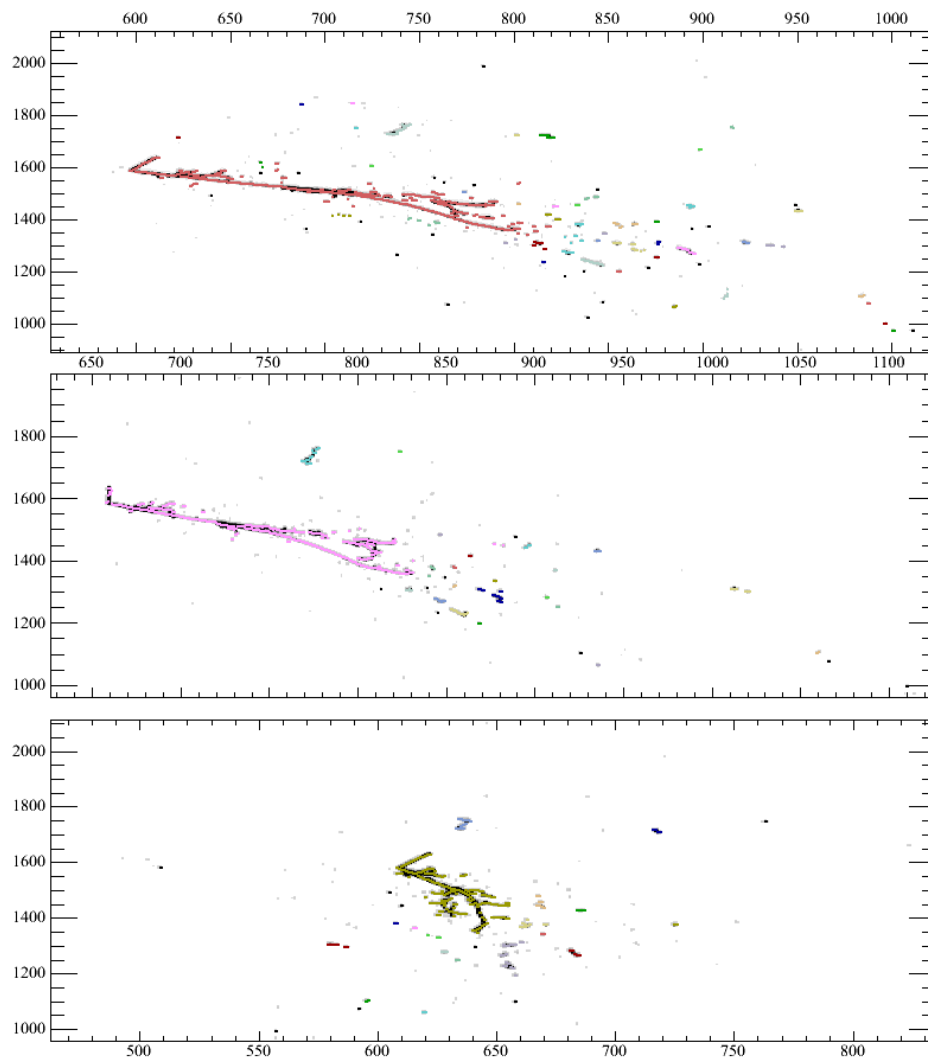
Fuzzy clustering



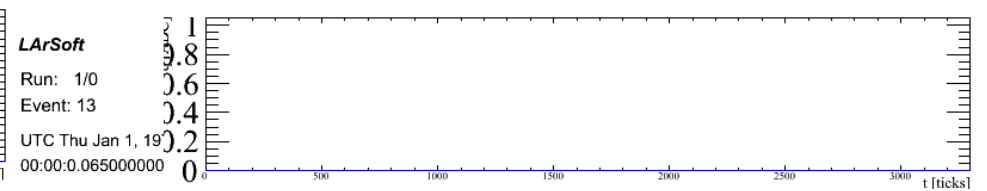
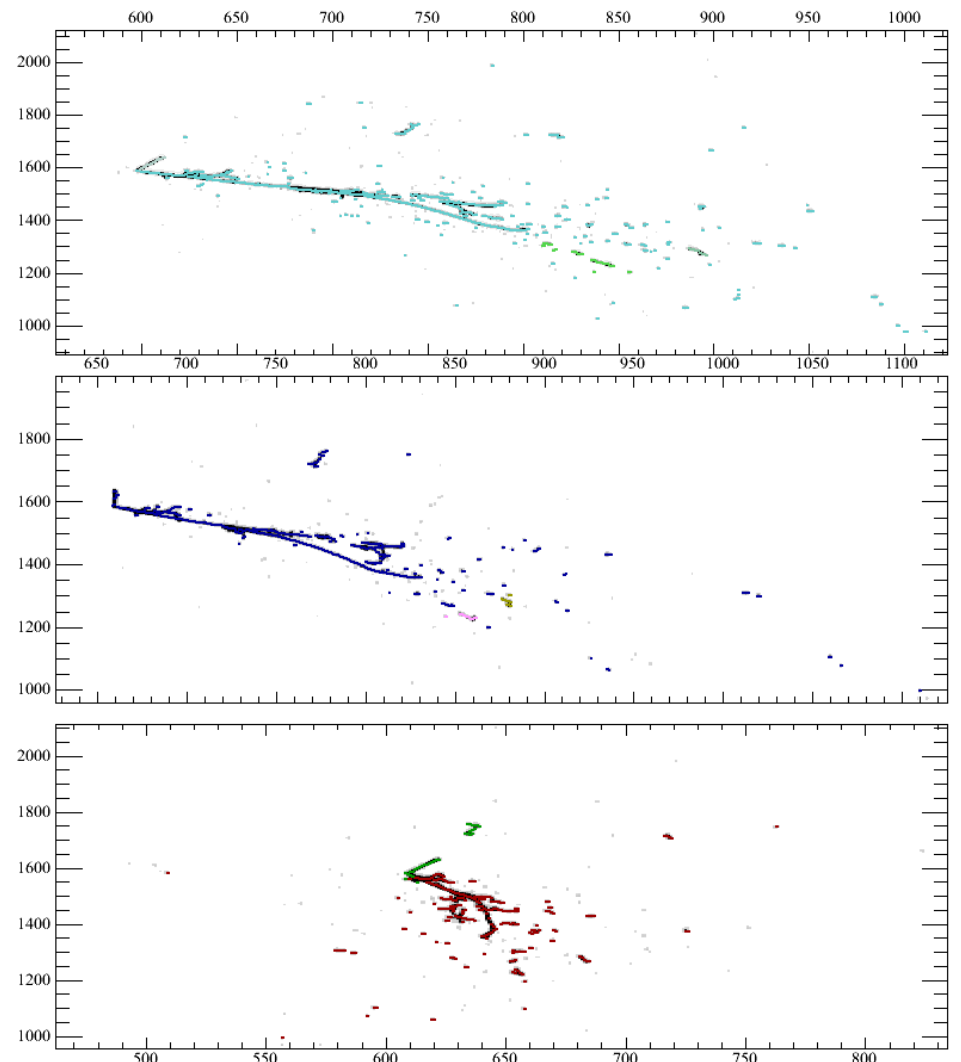
DBSCAN clustering



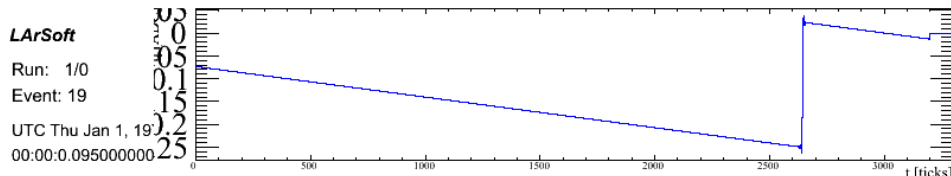
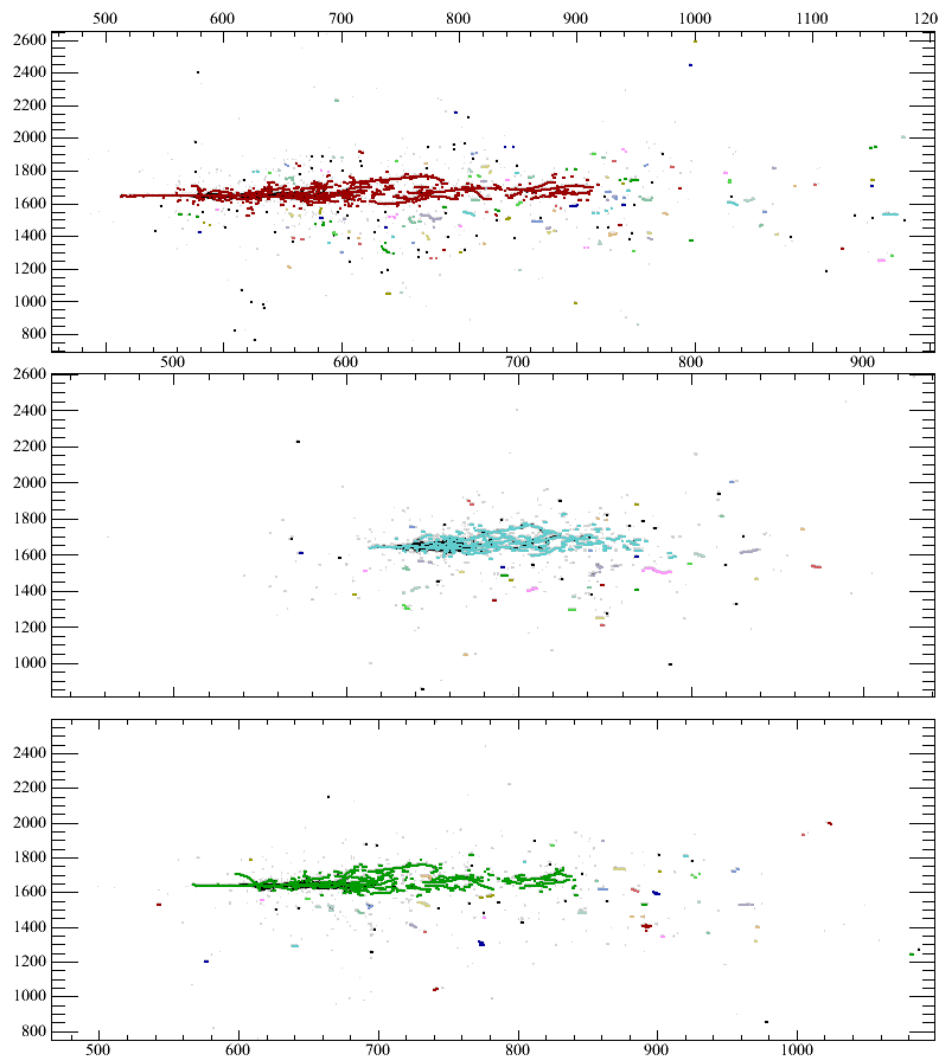
Fuzzy clustering



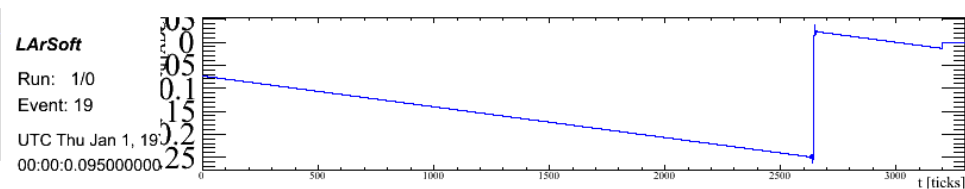
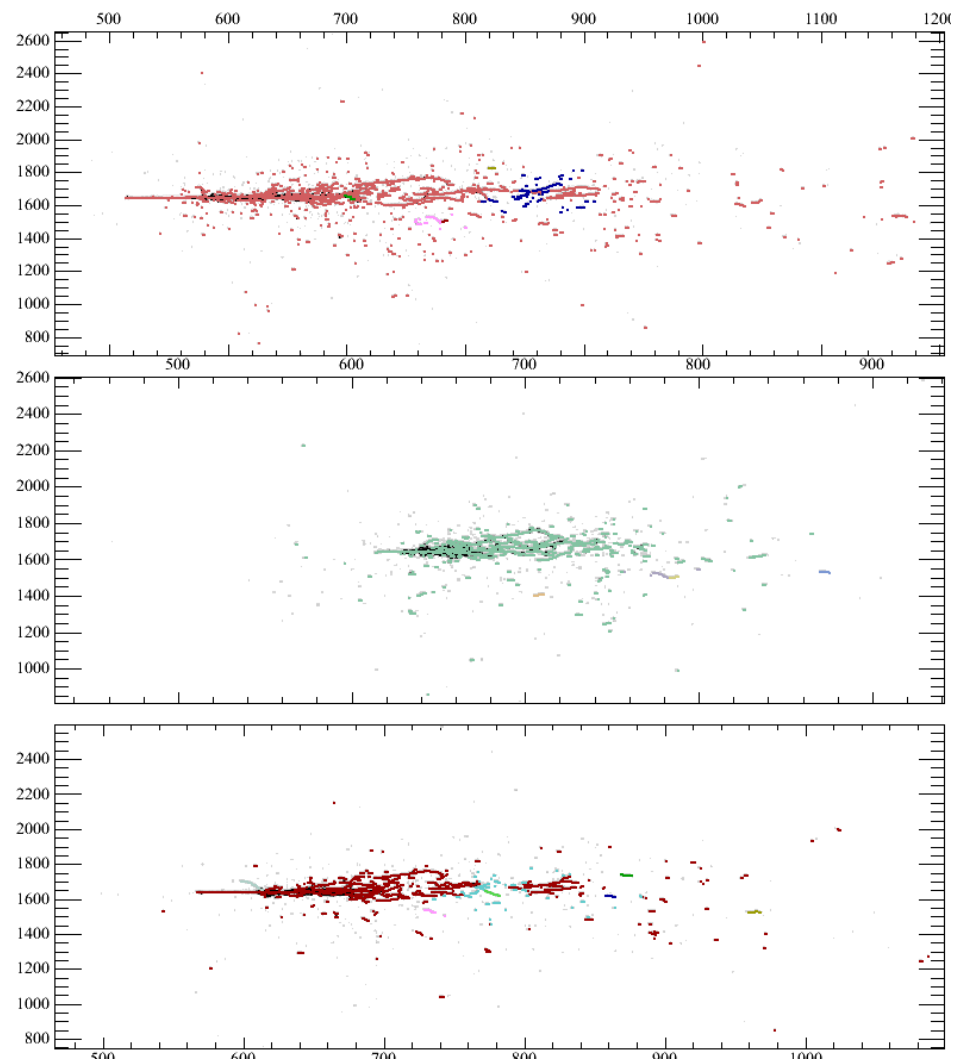
DBSCAN clustering



Fuzzy clustering



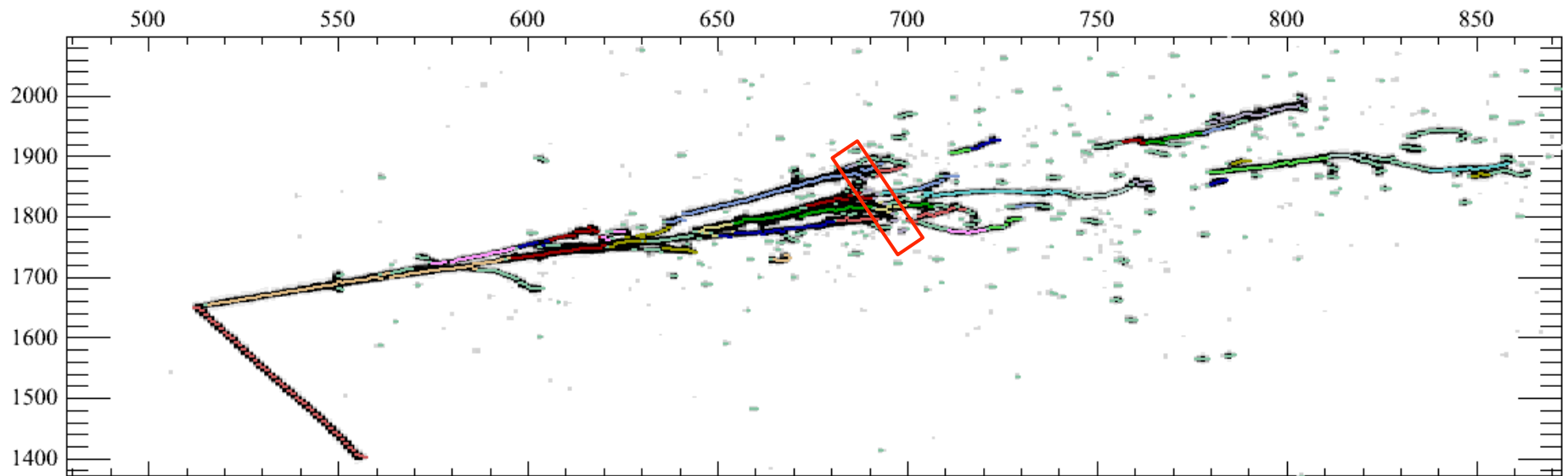
DBSCAN clustering



Fuzzy clustering

Fakes still an issue

- Strategy is to walk along the line and search for missing hits
- If too many missing hits are found, the line is vetoed



This is the event from slide 10, collection view; the fake line is a very faint grey; this is also the issue with the collection view on slide 14

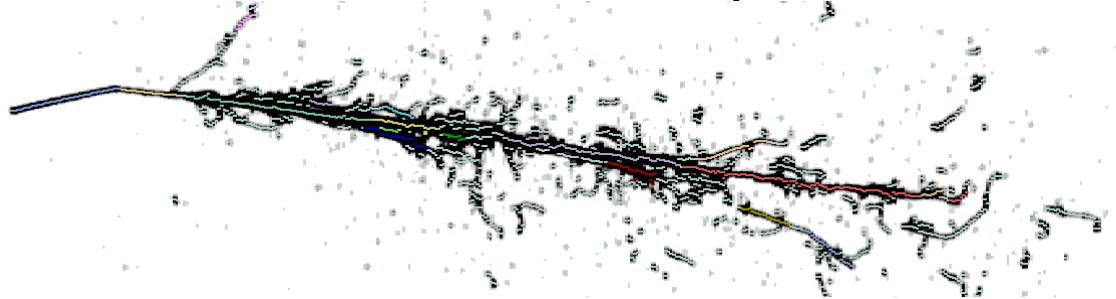
Other modifications

- Created a new class HoughBaseAlg that merged HoughClusAlg and HoughLineAlg
- This had the effect of slightly complicating the configuration file ClusterFinder/
clusteralgorithms.fcl
- Now have the ability to reproduce the PPHT using RandomNumberSaver, see Wiki (likely has other applications)

Back up

Status of Hough transform

- Effective at finding lines, yet very slow
- Fortunately, there are many avenues for potential improvements
- The biggest slowdown in the original HoughLineAlg resulted from refilling the accumulator from scratch after every line was found
- First optimization came from subtracting points from the original accumulator after each point was found



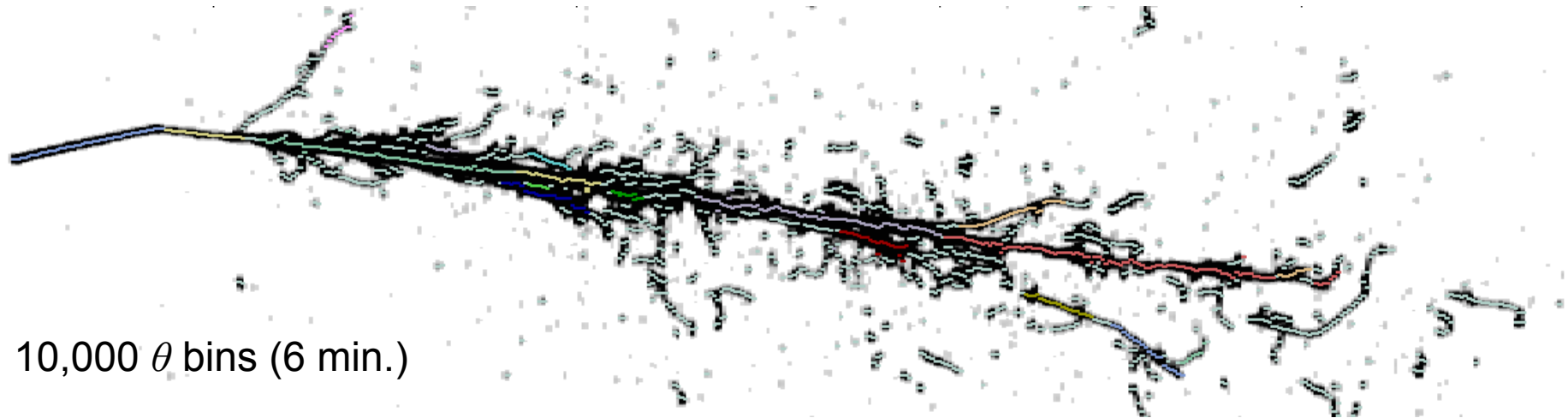
For this event, took processing time from 37 minutes to 6 minutes, even running at double the θ resolution

Going even faster with a PPHT

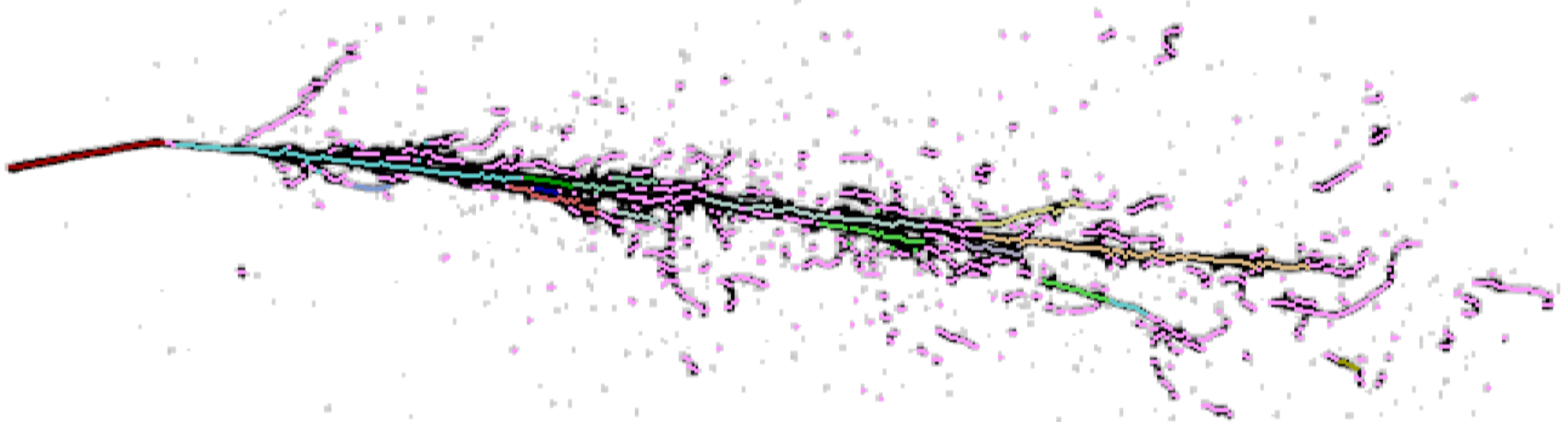
- Early work using a Progressive Probabilistic Hough Transform (PPHT) has further increased speed
- The algorithm is fairly simple:
 1. Randomly select one hit, remove it from the image
 2. Check if the highest peak in the accumulator modified by the pixel is higher than a threshold cut; if not, go to step 1
 3. Add points along the line, removing them from the image
 4. Remove hits from the accumulator from the line that was found
 5. Repeat step 1 if the image is not yet empty

J. Matas et al., Robust Detection of Lines Using the Progressive Probabilistic Hough Transform, Computer Vision and Image Understanding, Volume 78, Issue 1, April 2000, Pages 119–137

PPHT performance



10,000 θ bins (6 min.)



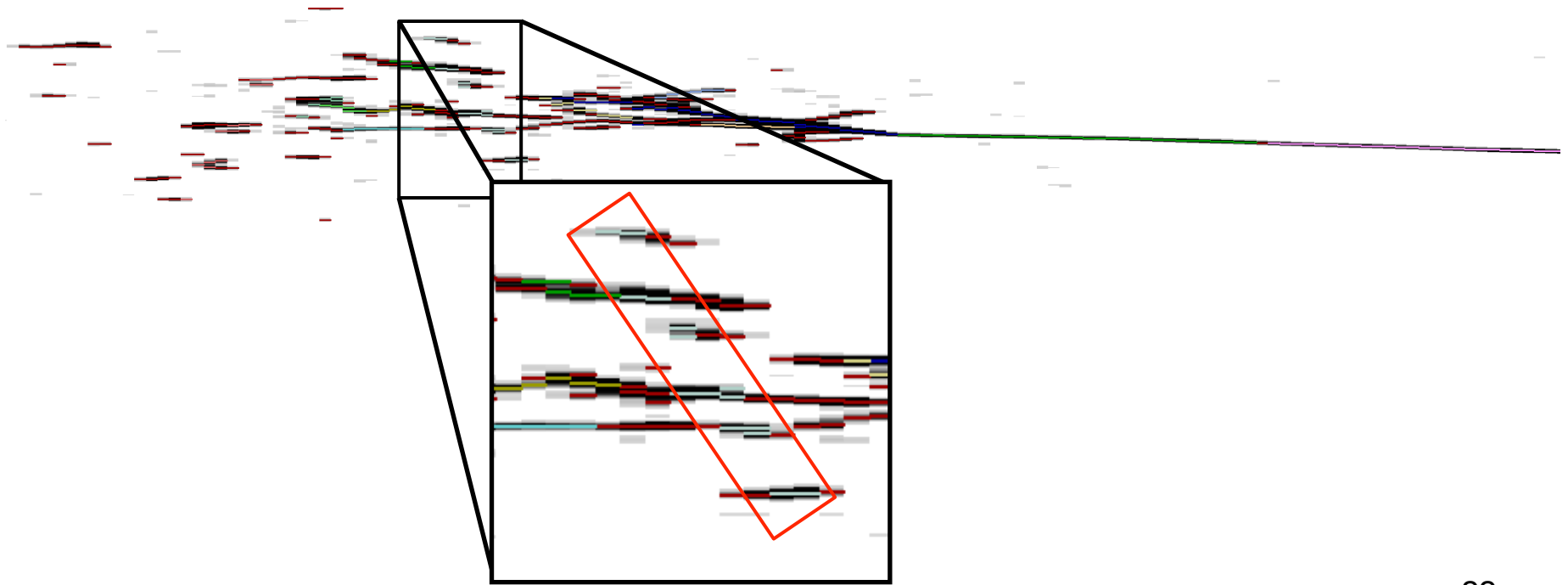
20,000 θ bins (1 min.)

Aim of the improvements

- Want to first apply a clustering algorithm (fuzzy clustering) for a rough first pass
- Then apply the PPHT to construct showers and tracks
- The PPHT is only implemented in the HoughClusAlg class, usable by fuzzy clustering code only, could be implemented in HoughLineFinder

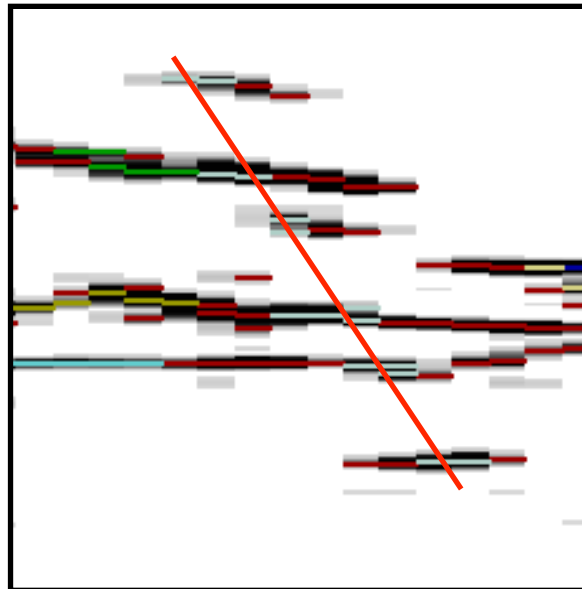
Dealing with fake lines

- The Hough line finders tend to create lines out of steep, uncorrelated hits



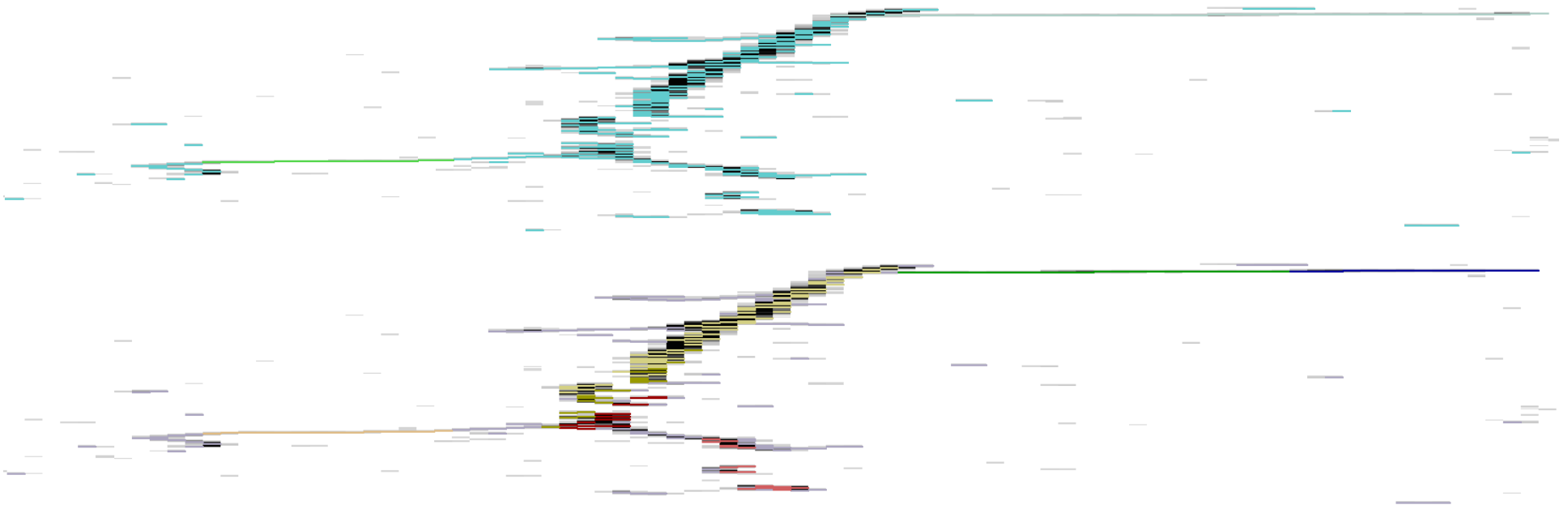
Dealing with fake lines

- Strategy walks along the line and searches for missing hits
- If too many missing hits are found, the line is vetoed



Dealing with fake lines

- Still needs some slight tuning, occasionally rejects good lines
- Most of the lines in this shower were rejected



Hough transform

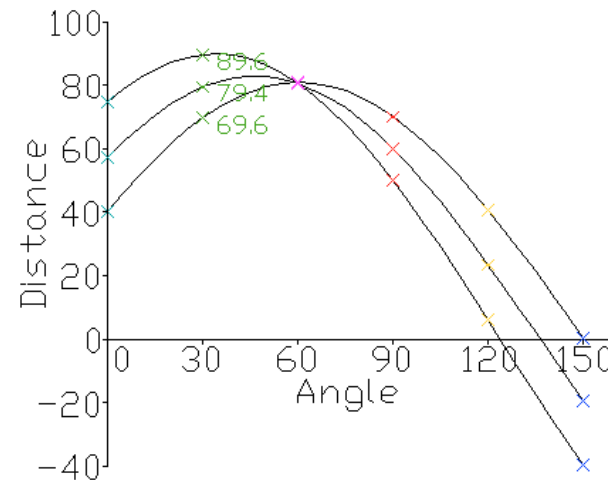
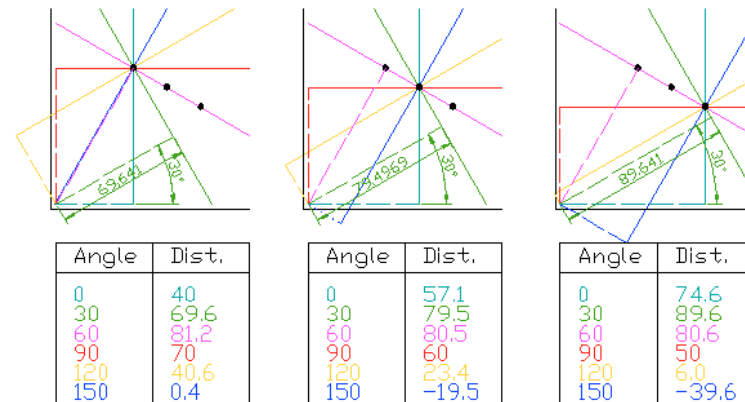
Use the parameterization:

$$y = \left(-\frac{\cos \theta}{\sin \theta} \right) x + \left(\frac{r}{\sin \theta} \right)$$

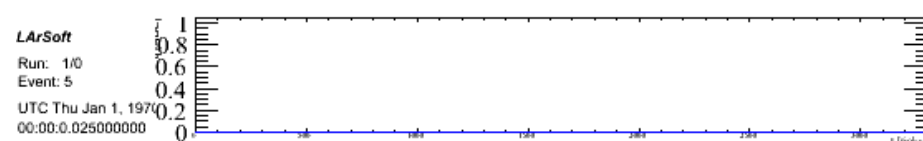
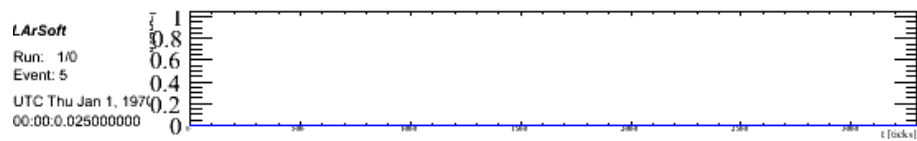
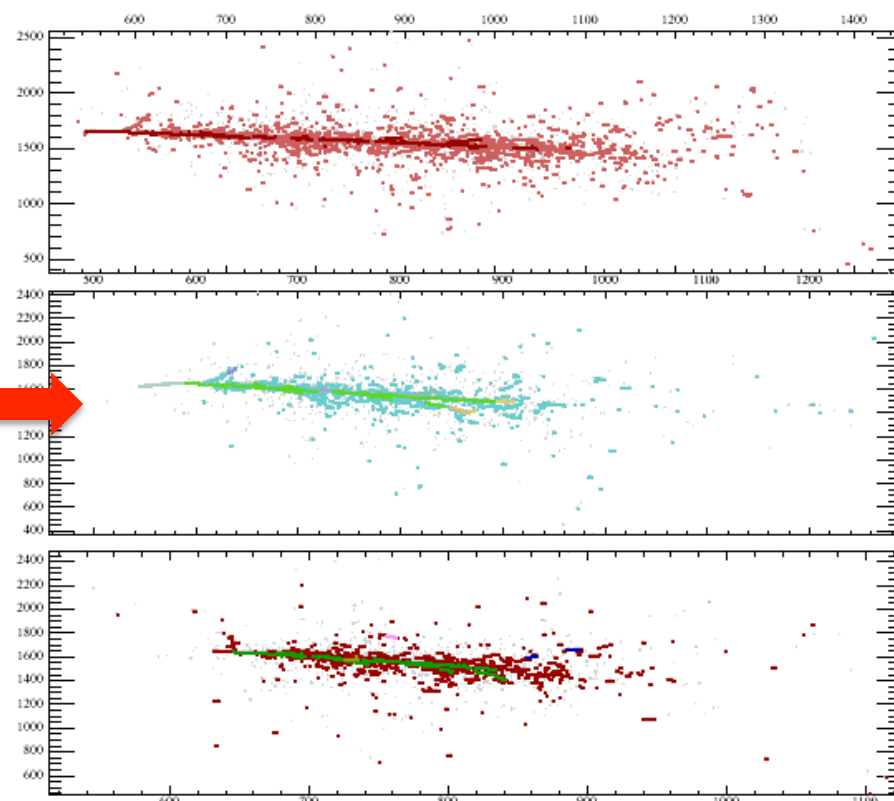
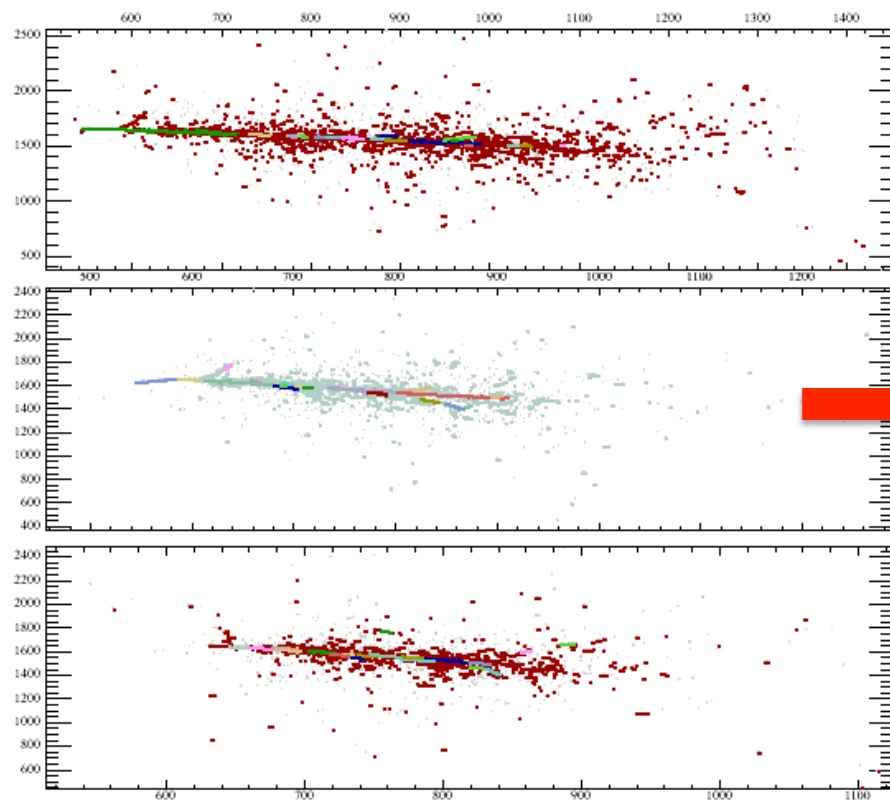
Fill a matrix (accumulator) in (r, θ) space with:

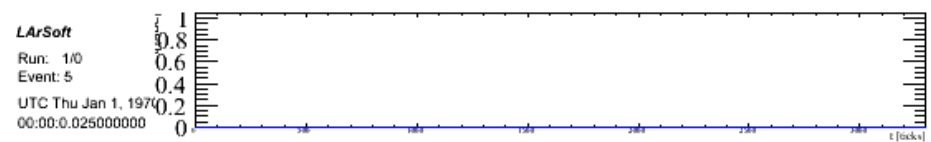
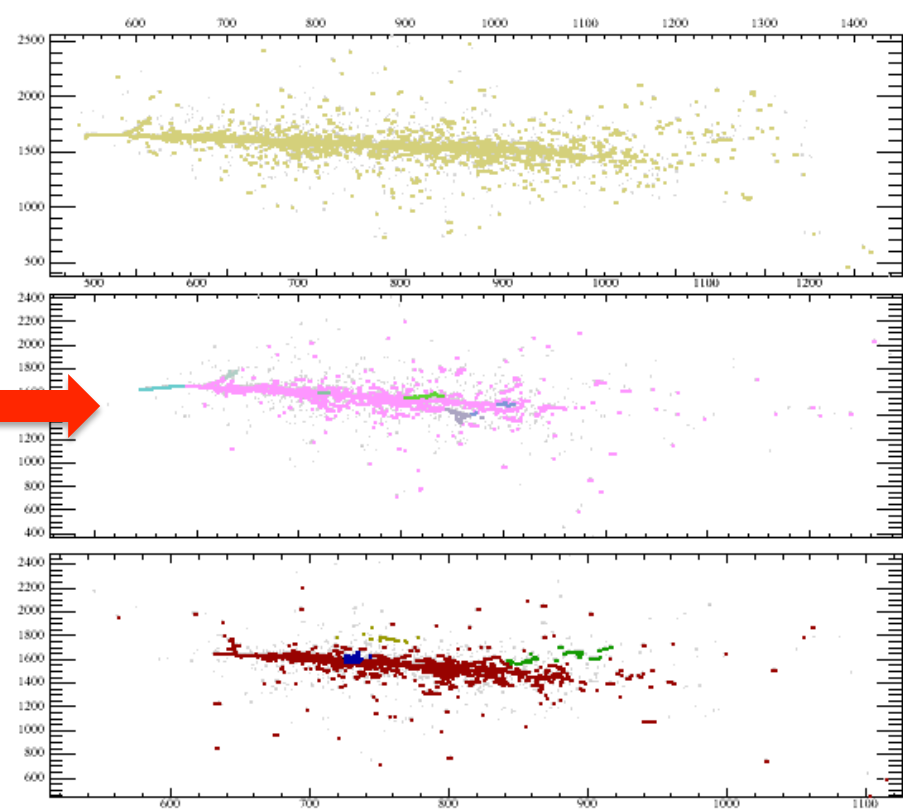
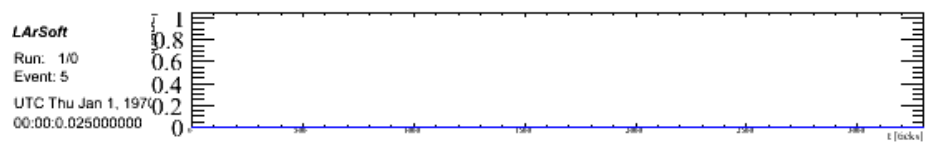
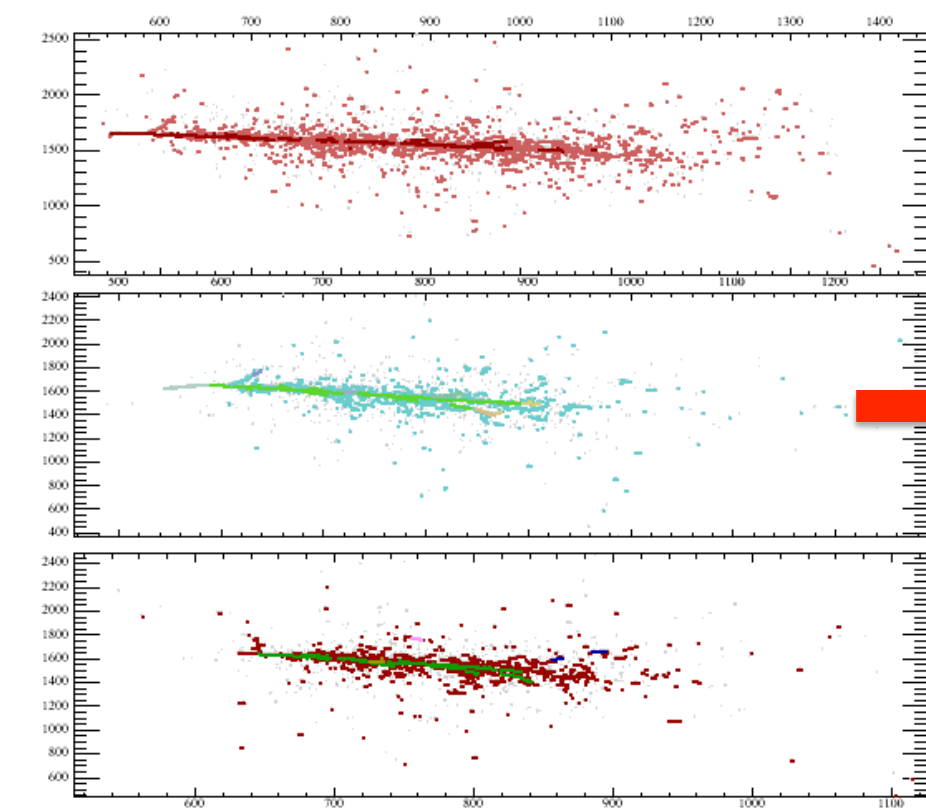
$$r(\theta) = x_0 \cdot \cos \theta + y_0 \cdot \sin \theta$$

We then search the accumulator for the most populated bin, rather computationally demanding



The accumulator, binned





Where clustering was

- Want to finalize clustering occurring right after hit reconstruction
- Lots of tools already available (e.g. DBSCAN, Hough line finder, and End Point Finder)
- Goal is to bring them all together

The basic algorithm

1. Initialize $U=[u_{ij}]$ matrix, $U^{(0)}$
2. At k -step: calculate the centers vectors $C^{(k)}=[c_j]$ with $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^N u_{ij}^m \cdot x_i}{\sum_{i=1}^N u_{ij}^m}$$

3. Update $U^{(k)}, U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^C \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4. If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$ then STOP; otherwise return to step 2.

The objective function we are trying to minimize:

$$J_m = \sum_{i=1}^N \sum_{j=1}^C u_{ij}^m \|x_i - c_j\|^2$$

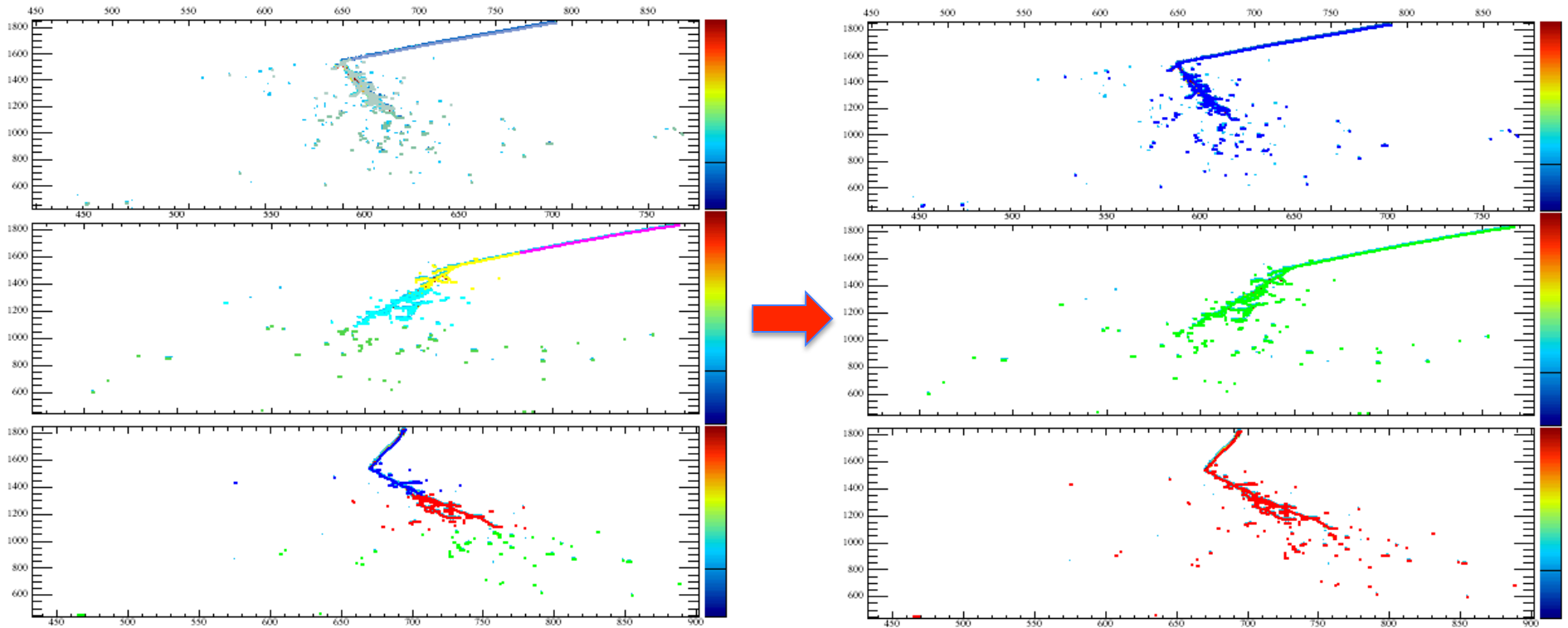
Number of clusters

- The number of clusters needs to be given as an input
- The Xie-Beni index gives us a way to evaluate how well a certain cluster number works

$$\frac{\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K u_{ik}^m \|x_i - c_k\|^2}{\min_{k,\ell} \|c_k - c_\ell\|^2}$$

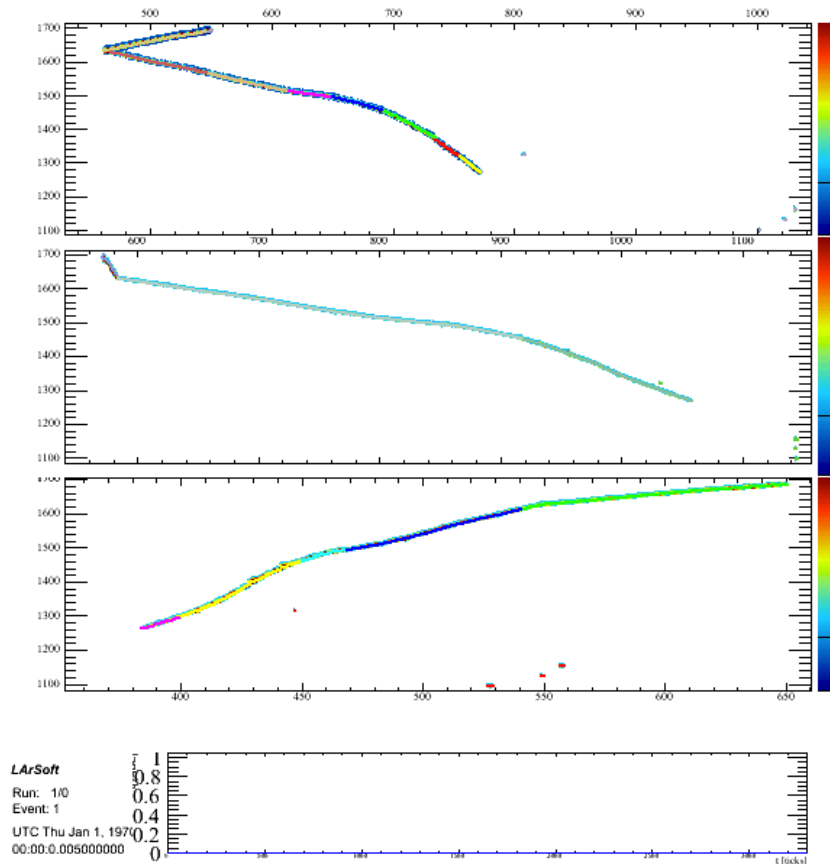
Does not work with single cluster events by definition,
we'll get to that later

Look to merge clusters using Euclidean distance



1. Look for point in cluster i closest to centroid in cluster j
2. Compare that point to the point in the cluster j closest to centroid of cluster i
3. Merge the clusters

Now look for Hough lines and expand them into clusters

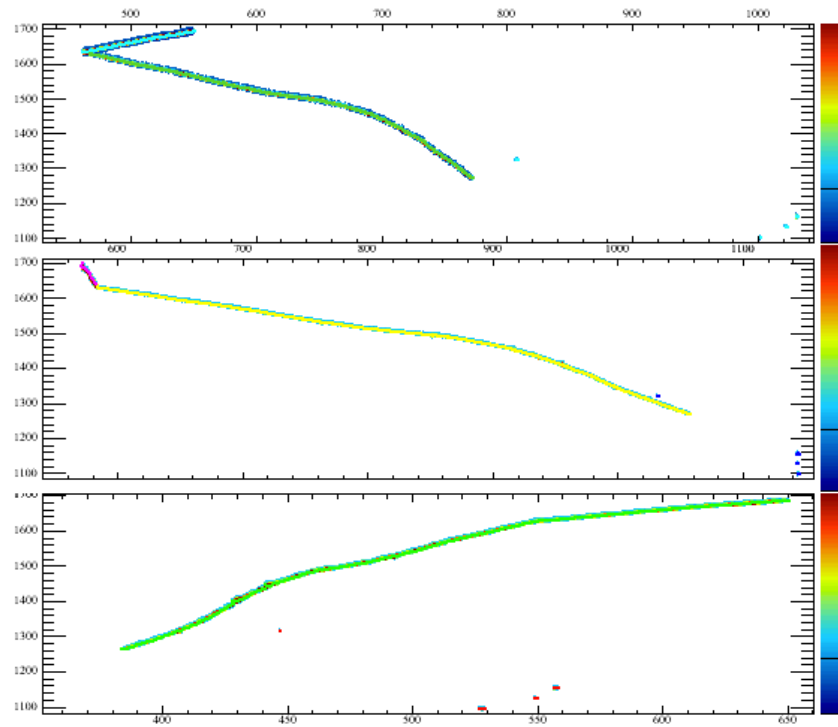


Search for Hough lines in the already identified fuzzy clusters

Hough line finder was sped up to make this practical

An issue remains with needing to merge the Hough lines though

Merging the Hough lines



Pick a line and check at its end points

If a nearby line is found, check the angle between the slopes

If the angle between the slopes is $< 30^\circ$, merge the lines

